Open Research Online

The Open University's repository of research publications
and other research outputs

Loosely coupled web representations: a REST service and JavaScript wrapper for sharing web-based visual representations

Conference or Workshop Item

How to cite:

Collins, Trevor; Quick, Kevin; Joiner, Richard and Littleton, Karen (2013). Loosely coupled web representations: a REST service and JavaScript wrapper for sharing web-based visual representations. In: World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA) 2013, 24-27 Jun 2013, Victoria, Canada.

For guidance on citations see FAQs.

© 2013 Association for Advancement of Computing in Education

Version: Accepted Manuscript

Link(s) to article on publisher's website:
http://www.aace.org/conf/edmedia/submission/uploads/EdMedia2013/paper_3055_39957.docx

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's data policy on reuse of materials please consult the policies page.

oro.open.ac.uk

Loosely Coupled Web Representations: A REST Service and JavaScript Wrapper for Sharing Web-Based Visual Representations

Trevor Collins, Kevin Quick
Knowledge Media Institute
The Open University
United Kingdom
{trevor.collins|kevin.quick}@
open.ac.uk

Richard Joiner
Department of Psychology
University of Bath
United Kingdom
r.joiner@bath.ac.uk

Karen Littleton
Faculty of Education and
Language Studies
The Open University
United Kingdom
karen.littleton@open.ac.uk

Abstract: This paper presents the design and application of a web service architecture for providing shared access to web-based visual representations, such as dynamic models, simulations and visualizations. The Shared Representations (SR) system was created to facilitate the development of collaborative and co-operative learning activities over the web, and has been applied to provide shared group access to: a high-resolution image viewer, a virtual petrological microscope, and a forces and motion spring simulation. As well as explaining the architecture and three applications, we briefly present the findings from a user study looking at primary school children's use of a shared spring simulation. The study findings indicate that shared access to a web-based simulation complements exploratory discussion and enables learners to demonstrate their understanding of a subject. Future work will explore how the system can be combined with dialogic support and embedded in activities to encourage learners with contrasting opinions to discuss and resolve their differing perspectives.

Introduction

Research on collaborative (and co-operative) learning has shown the importance of particular forms of talk for collective reasoning and learning. Productive talk depends on children's communications skills and communication is known to be multimodal and situated. Pea (1993) writes about the social and material embeddedness of communication:

"Conversations and interactions in everyday life take place in a rich referential field. The dense texture of human bodily orientation, gesture, and facial expression are known to communicate and continually transform on a moment-to-moment basis affective, cognitive, and social dimensions of relationships. Just as profoundly, there is a material environment to which attention can be directed, by gaze, pointing, and other means, in this conversational space. It, too, is transformed on a moment-to-moment basis. This material environment certainly includes physical objects, but it is also likely to include external representations, or inscriptions, such as writing and sketches, and in more formal settings, whether in school or work, such symbolic artifacts as equations, diagrams, maps, and designs." (Pea, 1993, p. 286)

Pea's description of everyday talk draws attention to the role of the material environment for focusing attention and supporting communication through the appropriation of external representations. This paper develops a system for sharing access and control of web-based representations (e.g. models, simulations and visualizations) that can be used by members of a group to facilitate and structure discussion within dialogic learning activities.

Conversations for learning

This research draws on Mercer's sociocultural discourse analysis, which distinguishes between disputational, cumulative and exploratory talk (Littleton & Mercer, 2013; Mercer & Littleton, 2007; Mercer, 2007). Disputational talk is characterized by disagreements, by individualized decision-making, and by short confrontational interchanges between speakers. It is often associated with competitive behavior and poor

learning outcomes. Cumulative talk involves speakers in friendly discourse with positive, but uncritical, exchanges that build on a common understanding through repetition, confirmation and elaboration. It is often useful at certain stages of the discussion (e.g. at the start). Exploratory talk, which is most clearly associated with productive learning, represents a social mode of thinking – so called 'interthinking' (Littleton & Mercer, 2013) involving constructive engagement with other people's ideas, based on reasoned justification and explicit consideration of alternative views. It draws on Mercer's notion of a shared dynamic dialogic space, which is the focal point of children's collective reasoning and co-construction of knowledge (Mercer, Hennessy, & Warwick, 2010).

Web-based dynamic representations

Web-based representations (e.g. models, simulations and visualisations), like other dynamic representations, provide a material object that can be used to help develop a practical awareness of a topic, but with the advantage that multiple people can access the representation over the web using a standard browser. In order to enhance communication within groups, this paper proposes a system for sharing access and control of web-based representations. The approach is intended to provide an opportunity for users to work together on the same representation via their web browser, as part of a shared dynamic dialogic space (rather than working independently on identical simulations). It is argued here that shared representations can be useful in collaborative and co-operative activities that require the learners to demonstrate and explain their understanding of a topic, thereby, providing them with opportunities to validate and improve their understanding.

Recent developments in web standards offer the technical means for providing shared representations over the web. HTML5 better supports dynamic representations than previous HTML standards, as it introduces the canvas, line drawing and shape elements for creating visual representations (Berjon, Leithead, Navara Doyle, O'Connor, & Pfeiffer, 2012) and makes extensive use of within-page JavaScript calls to support dynamic interaction (Auborg, Song, & Steen, 2012). Furthermore, development libraries, such as JQuery and Tonic, provide layers of abstractions that facilitate interface and service development. These improvements have enabled web designers and developers to create visually engaging and interactive web applications (e.g. http://html5-showcase.com/).

Loosely coupled systems

The concept of a loosely coupled system generally refers to a system that is comprised of independent parts (Glassman, 1973). This has been a useful concept for discussing organizational structures (Weick, 1976; Orton & Weick, 1990) and more recently, the concept has been adopted for describing desirable characteristics for web service architectures (Kaye, 2003; Pautasso & Wilde, 2009). In these examples, the loose coupling of related parts indicates that the parts of a natural or created system have minimal knowledge or awareness of one another and, therefore, can operate independently as part of the system.

The concept of a loose coupling is applied here to describe the relationship between the web server and the web clients (i.e. each web browser) used by a group of people (referred to as users). In a shared simulation, for example, the server holds the values of the simulation parameters, such as an object's weight. For the user controlling the weight of the object in the simulation, any changes they make are periodically sent to the server to update the value of the corresponding parameter (stored in a database). The simulation running in the other group members' browsers periodically check for and retrieve changes to the simulation's parameters from the web server. In this way, every client runs their own version of the simulation, but by sharing any changes made to the underlying parameters, the simulation running in each client behaves identically.

By setting the frequency of the periodic parameter checks to once a second, for example, every client will reflect a change within two seconds of the change being made (i.e. with a maximum delay of one second before the client currently in control updates the server, and a further second for each client to update their local value). This loose coupling enables the group size to be scaled to hundreds of users; although in practice multiple smaller-groups are more typically needed (e.g. a class of 30 pupils working in six groups, with four pupils in each group), and shorter update frequencies can be used if a quicker response is needed.

By using time to trigger the updating and checking of a representation's shared set of parameters, this approach can be used with any HTML5 page without modification, providing the JavaScript parameters used within the page can be identified and stored by the shared representation system.

Summary: Goal and basic approach

The goal of producing a system for sharing access and control over HTML5 pages is to enable a group of people to simultaneously explore a shared resource. This can be contrasted with accessing identical resources, which is how an HTML5 page is currently experienced. That is, when each user views a web page, their browser executes the JavaScript elements contained in the page, in order to give it a dynamic behavior, but the user's actions do not affect anyone else currently viewing the same page. In the case of visual representations, the user could dynamically change the selection of data sets and charts used for visualization, or the physical property values of a simulation. By storing the changes made by one user and reflecting that change in the web browsers of other group members' accessing the same page, the shared representation system effectively gives shared access to the same resource. The system could either be used by groups of co-located people or by geographically distributed groups. In both cases, the intended learning context would include the opportunity for verbal discussion either face-to-face or online (e.g. using VoIP telephony), as part of a collaborative and co-operative learning activity.

Design

This section explains the key design elements of the system that allows a group of users to share control of a web-based representation. First, the components of the system are introduced. The RESTful service that enables the system to create, read, edit and delete parameters using HTTP methods (i.e. web page requests) is then presented; and the code wrapper that associates the shared system with an existing HTML5 representation is described. Finally, the rationale and use of the three core data structures for managing groups of users, views and parameters are explained.

System architecture

The Shared Representation (SR) system is made up of a set of PHP, JavaScript and HTML5 files that are added to existing web-based representations on a web server, to provide a loosely coupled form of shared access and control over those representations for groups of users. Figure 1 illustrates the components of the SR system and their communication connections. The system can be accessed by any HTML5 compliant browser application (i.e. web client) running on phones, tablets, laptops or desktop computers. The SR system wrapper uses: the Tonic REST framework to provide a RESTful web service for handling parameter requests, the PHP scripting language for processing data, and a MySQL database for retrieving and storing data.

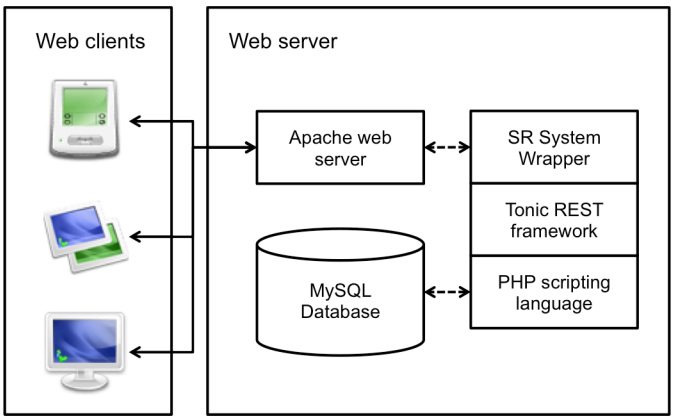

Figure 1. A schematic representation of the architecture used to implement the SR system.

RESTful service

REST stands for Representational State Transfer and is a set of architectural principles for designing web services, originally proposed in Roy Fielding's PhD thesis (Fielding, 2000), which are now used widely (Richardson & Ruby, 2007; Rodriguez, 2008). The extent to which these principles are followed varies, but four core aspects are common, namely the use of: HTTP methods, stateless requests, structured web addresses (i.e. URIs), and simple data formats (e.g. XML and JSON).

The REST service used in the shared representations system is built using Tonic, a RESTful development library for PHP (James, 2009). HTTP methods are used for creating, reading, editing and deleting information stored by the system, specifically using the: POST, GET, PUT and DELETE methods (respectively). Each request is handled independently, so all of the context or state information needed to process a request is included with that request. This includes the user's group and authentication details. The path structure of the URIs is mapped to an intuitive structure for the system, so a specific representation is requested by name in the URI (i.e. http://hostname/[system]/[name]), and any actions associated with that representation are referred to by additional terms added to the path (i.e. http://hostname/[application]/[name]/[action]). Finally, the data returned from the service (e.g. the parameters and their values) is passed using the JavaScript Object Notation (JSON), a data interchange format developed by Douglas Crockford (2009). JSON represents data textually either as a collection of name/value pairs (e.g. {"firstName" : "John", "lastName" : "Smith"}) or as an ordered list of values (e.g. {"assignmentMarks" : ["61", "53", "64", 71"]}).

Code wrapper

A wrapper is used to associate the shared representation system with a HTML5 page. This includes the Tonic PHP library and the application-specific extensions, as well as a single file containing the page-specific properties. This customized property file includes the initialization and mapping functions used to: parse the JSON data returned from the server by a GET request into the data structure used by the HTML5 page; and to generate the JSON data sent to the server by POST and PUT requests from the data structure used by the HTML5 page.

The visual effect of the wrapper, is a control bar added to the page (see Figure 5). The control bar initially contains the title of the current view, a navigable list of available views, and a 'join' button; the page is unaffected otherwise and can be used as a single (i.e. not-shared) page. Once the user clicks on the join button and signs-in, they are part of a group and the control bar is updated accordingly. The control bar for a signed-in user toggles the state of the 'join' button to a 'leave' button, which the user clicks on to leave the group session. The control bar for an authenticated user also includes: a toggle button to take 'control' or 'release' control of the representation; the name of the signed-in user; a (handshaking) button to toggle the user's connection with the group; and a floating list of the signed-in group members, and their control and sharing status indicators.

Groups of users

Organizing users into groups and managing their membership for group activities can be quite challenging as the group membership may change at anytime and therefore need adjustment, for example, to accommodate absentees or changing social dynamics (Dillenbourg & Tchounikine, 2007). The approach taken here is to organize groups hierarchically, so that individual users can be associated directly with a group or indirectly through a subgroup (Collins, Mulholland, & Gaved, 2012). This enables the creation of grouping structures that can be used across a set of activities with a minimal amount of duplication. So, for example, a head of a school department might be associated with a group at the top of a user-grouping tree, which has a separate subgroup for each teacher within the department. The teacher-level groupings can then have corresponding subgroups for each class, and further class-level subgroups for each set of groupings used by the teacher with that class. The data structure used to represent user group hierarchies is illustrated in Figure 2.

Groups of users

users

u_id	*firstname*	*lastname*	*username*	*password*
u1	Ann	Smith	ann	…
u2	Bob	Smith	bob	
u3	Peter	Smith	peter	
u4	Paul	Smith	paul	

groups

g_id	*name*
g1	blue group
g2	red group
g3	…

group_user

g_id	*u_id*	*role_id*
g1	u1	…
g1	u1	
g2	u2	

group_hierarchy

g_id	*parent_g_id*
g1	
g2	g1
g3	g1

Figure 2. A schematic representation of the data model used for storing users accounts and organizing them hierarchically (see Figure 5 for an example application).

When a user accesses a shared representation, the URI they access it through includes an identifier for the representation and an identifier for the requested group-level. In this way, the same representation identifier can be used across groups. For example, if a class of pupils were working in pairs with a shared simulation each pair within the class would have a unique URI containing the common representation identifier and a distinct group-level identifier. The system uses the requested group-level to determine the group members by including signed-in users associated with that group, that group's subgroups, and groups along the path from that group to the top of the group tree. This hierarchical structure enables switching between regularly used groupings (such as pairs, groups of four, and so on), by modifying the URIs group identifier. Furthermore, the group level that a user is associated with is used in such a way as to minimize duplication across groups. So that, for example, a teacher can be associated at the level of a class rather than every subgroup within a class, but they can join any subgroup by using the corresponding group identifier.

Groups of views

In many contexts where representations are used to explore real-world data or processes, several related representations are involved. To support switching between representations, another hierarchical tree structure has been used for representations. In this case, when each shared representation is defined, identifiers are used to associate the representation with an immediate parent representation (if applicable) and the top (i.e. root) representation for that group of representations (see Figure 3). For implementation convenience, a group containing just one representation has no parent identifier and the root identifier refers to the representation itself.

Groups of views

shared_representations

sr_id	*parent_sr_id*	*root_sr_id*	*label*	…
sr1		sr1	earthworm	
sr2	sr1	sr1	fucus	
sr3	sr2	sr1	sycamore	
sr4		sr4	granite rock 1	
sr5	sr4	sr4	granite sample 1.1	

group_sr_view

g_id	*sr_id*	*current_view*
g1	sr1	sr3
g2	sr1	sr2
g3	…	

Figure 3. A schematic representation of the data model used for storing shared representations and organizing them hierarchically (see Figure 6 for an example application).

Using representation identifiers, a set of views can be navigated by links to the current view's immediate parent view and immediate sub-views. So, for example, a microscope slide of a rock sample could be associated with a parent view of the rock the sample was taken from, and sub-views showing specific areas of interest on the slide. As a user navigates from one view to another the identifier for the current view is stored by the system (for the corresponding pair of representation and group identifier values) and used to update the current view for each group member. In this way, the system enables the user that is currently in control to switch the view for everyone in the group. When a user joins a group and requests a view that is not the current focus of the group, the system will bring them directly to the view that is the current focus.

Groups of parameters

Only one person can be in control of a parameter at a time, but depending on the system being represented, the parameters of a representation can be grouped so that one person could control a set of parameters, which might include all the parameters or a subset of them. This is particularly useful in contexts where a learning activity is designed to create differences between group members to encourage discussion (Dillenbourg & Tchounikine, 2007). For example, in the case of a science simulation one user could change a property value, such as the speed of a component, and their partner could be given the task of inferring what was changed from observing the differences in the simulation's behavior. The data structure used to represent parameter sets is illustrated in Figure 4.

Groups of parameters

parameters					
sr_id	*g_id*	*name*	*value*	*control_group_id*	*controlled_by*
sr1	g1	x	1234		u1
sr1	g1	y	342		u1
sr1	g1	zoom	1.8		u1
…	…	…	…	…	…
sr7	g1	k	0.8	cg1	u3
sr7	g1	m	12	cg1	u3
sr7	g1	d	0.4	cg2	u2

parameter_control_groups		
cg_id	*sr_id*	*labels*
cg1	sr7	k+m
cg2	sr7	d
cg3	sr7	dr

Figure 4. A schematic representation of the data model used for storing sets of control parameters (see Figure 7 for an example application).

Within the system, where representation parameters are separated into more than one group, an ordered set of parameter labels are stored so that the corresponding set of toggle buttons can be added to the control bar for taking and releasing control over each group of parameters. When no parameter control groups are identified for a representation, the system presents a single toggle button (labeled control/release) for one user to take control or release control of the representation.

Applications

This section briefly illustrates how the SR system has been applied to provide shared group access to: a high-resolution image viewer, a (virtual) petrological microscope, and a forces and motion spring simulation.

Shared image viewer

The HTML5 Zoomify viewer is a widget for accessing high-resolution images as a collection of tiled images (see http://www.zoomify.com/html5.htm). A plug-in extension for image manipulation programs, like PhotoShop, can be used to transform a single high-resolution image into a collection of Zoomify image tiles. To create a shared image viewer, the JavaScript parameters of the HTML5 Zoomify viewer (i.e. x, y position and zoom level) were identified and used in the corresponding initialization and mapping functions of the SR property file, and a suitable set of user accounts and groups were created. Figure 5 shows the resulting application accessed by two users, one currently controlling the viewer (left) and the other watching (right). The actions of the user controlling the viewer will be shown in every user's browser, actions of other users (not in control) will be reverted by the system when it retrieves the stored set of parameters from the REST service.

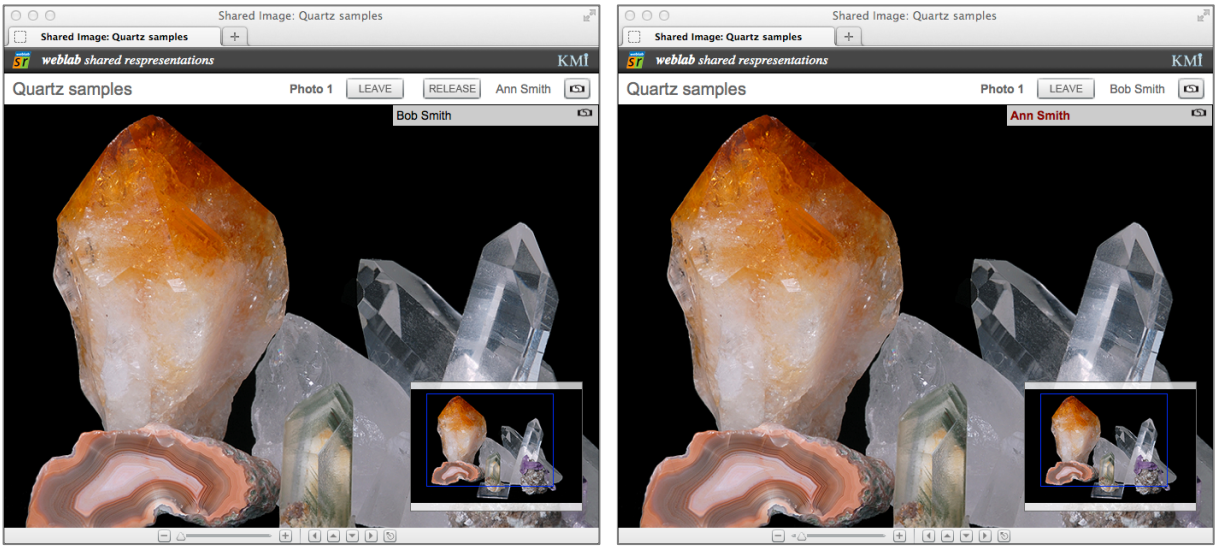

Figure 5. A shared Zoomify image viewer, showing screen views from two users in a group; one that is controlling the view (left) and one that is following (right).

Shared virtual petrological microscope

The virtual microscope application was configured in a similar way to the image viewer; the JavaScript parameters of the HTML5 virtual microscope were used to set up the property file. In this case, the application typically uses sets of representations, so a rock sample slide is associated with rotation point views at specific points of interest on the slide. Therefore, additional information was stored regarding the grouping of views when configuring each example. Figure 6 shows two associated views: a slide view (left) and one of three associated rotation views (right).

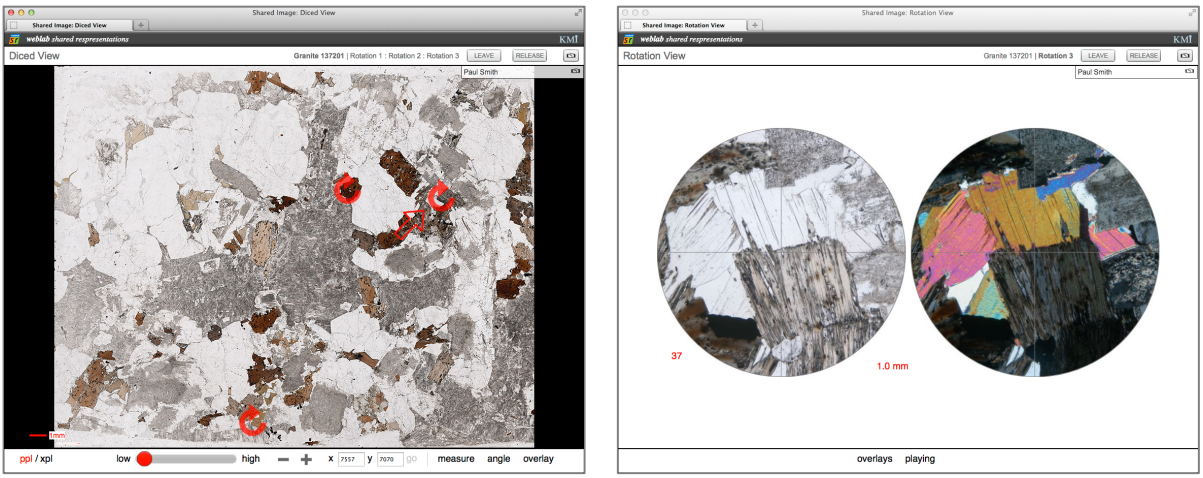

Figure 6. A shared petrological virtual microscope, showing a screen view of a slide containing multiple sub-views of interesting slide areas (left) and an example of the rotation view for one of the sub-views.

Shared forces and motion spring simulation

This application followed a similar configuration process for identifying the JavaScript parameters of a HTML5 spring simulation as described above. However, it also split the control parameters into distinct sets so that a user could take control (or release control) of: the spring constant and mass, the damping constant, or the speed of the driver motor. In the simulation, sliders on the left of the screen control parameters, the simulation is in the middle, and the behavior of the simulation is reported on the right hand side of the screen. As well as controlling the movement of the spring by dragging the motor slider, the user can drag the blue handle up-and-down to explore how the spring and mass behave. Figure 7 illustrates how two group members can take control of separate groups of simulation parameters.

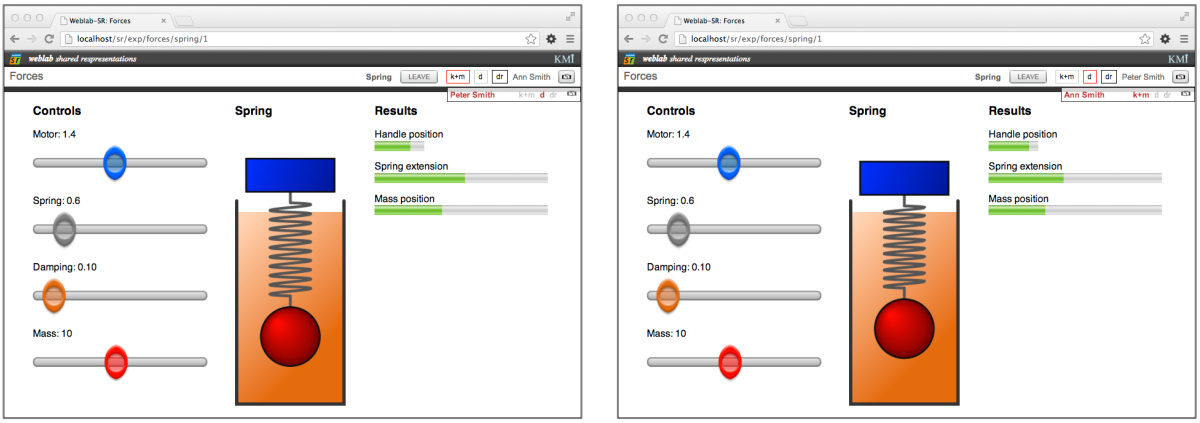

Figure 7. A shared forces and motion spring simulation, showing screen images from two users in a group, each in control of different sets of parameters (left = spring constant and mass; right = damping constant).

User study

The following study was carried out using an earlier prototype of the SR system to help validate the loosely coupled approach for sharing group access and control over web-representations. The findings presented here were used as part of a successful funding application that enabled the SR system to be developed further as described.

Method

The study involved 14 pairs of children from two primary schools in southern England (14 boys and 14 girls) with an average age of 10 years six months. Each child completed an individual pre-test, which consisted of eight questions. The first four questions were about a static spring, where either the mass or the strength of the spring was changed (referred to as static). The next two questions were about a static spring, where the mass and the strength of the spring were changed (referred to as inter). The children were shown a picture of a spring and asked what would happen, whether the spring would go down, go up or stay the same. The last two questions were on dynamic springs, where the spring was moving up and down and either the mass or the strength of the spring was changed (referred to as dynamic). The children were shown a picture of the spring and asked whether the weight would move up and down quicker, slower or stay the same.

After the pre-test, the children operated a shared spring simulation in pairs and worked through a scripted protocol. They each had an Apple iPad, which was wirelessly connected to a web server containing the shared representation system (see Figure 8). The protocol consisted of a series of questions where the children were asked to agree on what would happen to a static spring if the mass were changed and/or the strength of the spring was changed. They then changed the mass and/or strength of the spring and observed what happened. A further set of questions concerned what would happen if the mass and/or strength of the spring were changed when the spring was moving up and down, as with the previous set of questions the children had to collectively agree an answer before they could change the mass and/or spring strength and observe what would happen.

This protocol was based on the work of Howe and her colleagues (2007) on collaborative learning in science. After they worked in pairs on the shared spring simulation the children completed a post-test (which was the same as the pre-test). As well as comparing the pre and post test scores the activity was video recorded, but the analysis is not reported on here.

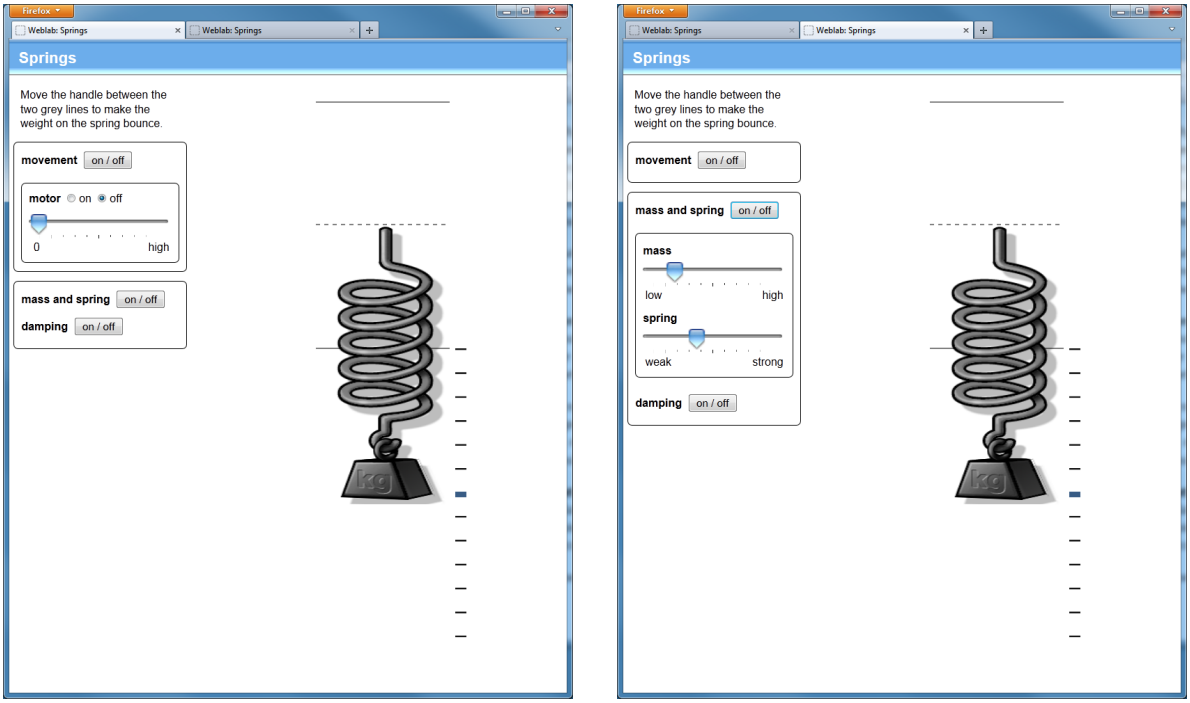

Figure 8. A shared spring simulation showing two different users (left and right) accessing the same simulation, but controlling different simulation parameters.

Results

Table 1 shows that the children had significantly improved on the: static problems (t=4.1, df=27, p=0.00033953), interactive problems (t=2.8, df=21, p=0.01), dynamic problems (t=2.9, df=21, p=0.01), and overall (t=5.7, df=27, p=0.00000468).

	Pre-test		Post-test		
	Mean	SD	Mean	SD	T
Static	2.8	1.3	3.8	0.7	4.1
Inter	0.9	0.9	1.4	0.7	2.8
Dynamic	1.1	0.5	1.5	0.6	2.9
Total	4.7	1.8	6.6	1.4	5.7

Table 1. Mean pre-test and post-test score from a study exploring primary school children's use of a shared spring simulation (in each case the improvement in scores was statistically significant at a level p <= 0.01).

Discussion

The significant improvements in test scores can be attributed to the use made of the shared simulation. However, it was not possible to have control groups in this study and therefore the use made of the shared simulation could not be compared to other approaches. Nonetheless, the results support the argument that using a shared representation as part of dialogic learning activity is beneficial. Further discourse analysis of the recorded sessions will help identify how the representation was used and the influence it had on the children's talk.

Conclusion and future work

This paper has presented the rationale, design and development of a software system for sharing access to web-based representations. The purpose of developing the system was to enable further investigation into the effective use of visual representations within groups, as a complementary support for exploratory discussion within dialogic learning activities. The key aspects of this generic system, which enable it to be applied flexibly across domains and levels of education, are: the adoption of a RESTful web service for managing loosely coupled representations; the wrapping of the system with (existing) widgets using a customizable code wrapper; and the hierarchical structuring of groups, views and control parameters. One of the applications of this system has been found to improve primary children's understanding of gravitational forces and motion when used by children working in pairs.

Future work will continue to develop the SR system through refining the implementation, informed by user trials and new applications. Additional studies are planned to explore how shared representations can be embedded within dialogic learning activities, and how the logged activities of each user can be appropriated to guide group formation and provide effective scaffolding for the user.

References

Auborg, J., Song, J., & Steen, H. (2012). *W3C XMLHttpRequest Web Standard Specification*. Retrieved from http://www.w3.org/TR/XMLHttpRequest/

Berjon, R., Leithead, T., Navara Doyle, E., O'Connor, E., & Pfeiffer, S. (2012). *W3C HTML 5.1 Web Standard Specificatioin: A vocabulary and associated APIs for HTML and XHTML*. Retrieved from http://www.w3.org/TR/html51/

Collins, T. D., Mulholland, P., & Gaved, M. B. (2012). Scripting Personal Inquiry. In K. Littleton, E. Scanlon, & M. Sharples (Eds.), *Orchestrating Inquiry Learning*. Routledge.

Crockford, D. (2009). *Introducing JSON*. Retrieved from http://json.org/

Dillenbourg, P., & Tchounikine, P. (2007). Flexibility in macro-scripts for computer-supported collaborative learning. *Journal of Computer Assisted Learning*, *23*(1), 1–13.

Fielding, R. T. (2000). *Archtiectural Styles and the Design if Network-based Software Archtectures* (PhD Thesis). University of California, Irvine, Illinois, USA. Retrieved from http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Glassman, R. B. (1973). Persistence and loose coupling in living systems. *Behavioral Science*, *18*(2), 83–98. doi:10.1002/bs.3830180202

Howe, C., Tolmie, A., Thurston, A., Topping, K., Christie, D., Livingston, K., … Donaldson, C. (2007). Group Work in Elementary Science: Towards Organisational Principles for Supporting Pupil Learning. *Learning and Instruction*, *17*(5), 549–563.

James, P. (2009). *Tonic - The RESTful Web App PHP Micro-Framework*. Retrieved from https://github.com/peej/tonic

Kaye, D. (2003). *Loosely Coupled: The Missing Pieces of Web Services*. RDS Press.

Littleton, K., & Mercer, N. (2013). *Interthinking: Putting Talk to Work*. Routledge.

Mercer, N. (2007). Sociocultural discourse analysis: analysing classroom talk as a social mode of thinking. *Journal of Applied Linguistics*, *1*(2), 137–168. doi:10.1558/japl.v1i2.137

Mercer, N., Hennessy, S., & Warwick, P. (2010). Using interactive whiteboards to orchestrate classroom dialogue. *Technology, Pedagogy and Education*, *19*(2), 195–209. doi:10.1080/1475939X.2010.491230

Mercer, N., & Littleton, K. (2007). *Dialogue and the Development of Children's Thinking: A Sociocultural Approach* (New edition.). Routledge.

Orton, J. D., & Weick, K. E. (1990). Loosely Coupled Systems: A Reconceptualization. *The Academy of Management Review*, *15*(2), 203. doi:10.2307/258154

Pautasso, C., & Wilde, E. (2009). Why is the web loosely coupled?: a multi-faceted metric for service design. In *Proceedings of the 18th international conference on World wide web* (pp. 911–920). New York, NY, USA: ACM. doi:10.1145/1526709.1526832

Pea, R. D. (1993). Seeing What We Build Together: Distributed Multimedia Learning Environments for Transformative Communications. *The Journal of the Learning Sciences*, *3*(3), 285–299. doi:10.2307/1466823

Richardson, L., & Ruby, S. (2007). *RESTful Web Services* (1st ed.). O'Reilly Media.

Rodriguez, A. (2008). *RESTful Web services: The basics* (CT316) (p. 11). IBM. Retrieved from http://www.ibm.com/developerworks/webservices/library/ws-restful/

Weick, K. E. (1976). Educational Organizations as Loosely Coupled Systems. *Administrative Science Quarterly*, *21*(1), 1. doi:10.2307/2391875

Acknowledgements

The authors would like to acknowledge Peter Whalley for his inspirational work on web simulations that lead to development of the SR system, and thank him for the support and help he gave us with the primary school study. We would also like to thank Jon Linney for his invaluable graphic design support and advice. The development of the RESTful service and JavaScript library was funded by the OpenScience Lab grant from the Wolfson Foundation (http://www.open.ac.uk/blogs/openscience/).